AFRL-IF-WP-TM-2002-1563

VHSIC HARDWARE DESCRIPTION LANGUAGE (VHDL) 200X REQUIREMENTS REPORT/SURVEY

James Aylor, University of Virginia
Robert Klenke, Virginia Commonwealth University
Ron Waxman, EDA Consulting
Paul Menchini, Menchini & Associates
Jack Stinson, Advanced Technology Institute
Bill Anderson, Advanced Technology Institute

Advanced Technology Institute
5300 International Boulevard
Charleston, SC 29418

NOVEMBER 1999

Final Report for 14 September 1998 – 22 November 1999

20020926 059

INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7334

NOTICE

MICHAEL T. MILLS
Project Engineer
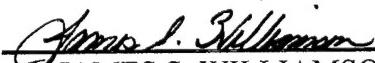
JAMES S. WILLIAMSON, Chief
Embedded Info Sys Engineering
Information Technology Division
Information Directorate
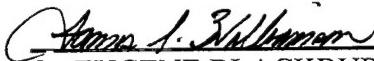
for EUGENE BLACKBURN, Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
November 1999	Final	09/14/1998 – 11/22/1999

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
VHSIC HARDWARE DESCRIPTION LANGUAGE (VHDL) 200X REQUIREMENTS REPORT/SURVEY	F33615-98-C-1351
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62204F

6. AUTHOR(S)	
James Aylor, University of Virginia	**5d. PROJECT NUMBER** 6096
Robert Klenke, Virginia Commonwealth University	
Ron Waxman, EDA Consulting	**5e. TASK NUMBER** 40
Paul Menchini, Menchini & Associates	**5f. WORK UNIT NUMBER** 3B
Jack Stinson, Advanced Technology Institute	
Bill Anderson, Advanced Technology Institute	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Advanced Technology Institute 5300 International Boulevard Charleston, SC 29418	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY ACRONYM(S)
Information Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson AFB, OH 45433-7334	AFRL/IFTA
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-IF-WP-TM-2002-1563

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Due to lack of funding, work on the contracted effort was terminated. This is the only document that resulted from this contract, and this is the best quality available. The latest VHDL standard (IEEE Std 1076) can be obtained through the IEEE.

14. ABSTRACT

The original objective of the VHDL-200X project was to provide various mechanisms to update the VHSIC Hardware Description Language (VHDL) Language Reference Manual (LRM) for the year 2000 and to ensure the successful standardization of that update. The contractual effort was to include research and collaboration with the Air Force and other VHDL users and tool vendors to extend and enhance VHDL to meet current day user requirements.

Since the originally planned 3-year effort was cut short to 1 year of funded effort, this report is submitted to document results of work accomplished during the 1 year of effort. It documents the results of the survey of the VHDL user and vendor community that were used to formulate requirements for updates to the VHDL standard.

15. SUBJECT TERMS
VHDL, VHSIC Hardware Description Language, VHDL-200X

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	70	Michael T. Mills **19b. TELEPHONE NUMBER** *(Include Area Code)* (937) 255-6548 x3583

HES&S 31-15093-1

Table of Contents

THIS PAGE WAS INTENTIONALLY LEFT BLANK

1   Introduction

This report was prepared by the VHDL 200x team, which is sponsored by the Air Force Research Laboratory. The work is being done in cooperation with the IEE DASC VHDL Analysis and Standards Group (VASG). Team members include: Jim Aylor of the University of Virginia, Bob Klenke of Virginia Commonwealth University, Paul Menchini of Menchini & Associates, Ronald Waxman of EDA Standards Consulting, Jack Stinson and Bill Anderson of the Advanced Technology Institute.

This is the first part of a two-part report. Phase I has accomplished the goal of gathering general information from a large population of users to help compile a list of user needs. The needs identified will be used during Phase II to formulate specific requirements for the next update to the VHDL language (VHDL 200x). Results of Phase I are being used to direct the areas of concentration for the VHDL 200x project as the second phase of requirements gathering gets underway. The VHDL 200x team will conduct detailed interviews with cognizant VHDL individuals within the VHDL community (users, tool builders, and researchers) to help gather specific requirements for the language update.

The results of Phase II will enable the VHDL language development community (the VASG) to make a significant enhancement in the capabilities of VHDL for the year 200x, well beyond the "maintenance release" which is in process today. By ensuring the buy-in of the tool producers as well as the tool and language users, the VHDL 200x revision will dramatically reduce the impact of electronic system obsolescence and pave the way for future generations of CAD/EDA systems that support rapid and cost-effective development.

This report contains three sections and three appendices. Section 1, the Introduction, provides an overview of the report. Section 2 presents the analysis of the recent user survey conducted by the VHDL 200x team. The analysis provides the guidance that will lead the Team in its quest for solid language requirements. Section 3 details the next steps for the team (Phase II). During Phase II, the team will be interacting with 1) the VASG to refine the needs expressed in Phase I, 2) cognizant VHDL individuals to develop specific recommendations for language requirements, and concurrently, 3) the VASG and the Air Force sponsors, to examine the proposed requirements for final inclusion in the language evolution process. The results of Phase II will be contained in the second part of the VHDL 200x report.

Appendix 1 contains the charts presented in the VHDL 200x workshop session in the October 1999 VIUF Conference that describes the study and provides a preliminary analysis of the data captured at that point in time. Appendix 2 contains the compiled raw data captured in Phase I of the study, except for the essay responses, which are quite lengthy and can be found at www.vhdl200x.org. Appendix 3 describes the survey methodology and contains the survey form.

2   User Community Guidance for Requirements Development

The requirements depicted below are "soft" requirements, in the sense that they provide guidance for the development of "hard" or specific language requirements. Specific

language requirements will be stated in a way that aids the language developer and tool builders to implement the recommended requirements.

The survey results were broken down into the major categories described below. In each section, discussion of the data received yields a preliminary analysis of that data. The reader is referred to Appendix 2 for the compiled raw data.

2.1 Demographics (Appendix 1 - charts 6-10)

This section of the survey included questions on the respondent's major job description, their companies' product area, and their use of HDLs. From the data, it appears that the majority of the respondents are VHDL users, but not Verilog users. This is somewhat expected from the distribution of the survey announcement. We were expecting more of a mix in that most tools today support both languages in cosimulation. The assumption for a mix is the belief that the user community is fairly bilingual and must work with systems where components are modeled in VHDL and Verilog. Perhaps the low percentage of Verilog users can be explained by the focus of the survey – primary Verilog users did not feel compelled to answer questions on a VHDL survey. One very interesting result is the significant percentage of respondents who claimed to be using C or C++ as a hardware description language. It is not clear if this use is in modeling hardware itself or in modeling the software that is a major part of most embedded systems being currently developed. Further investigation of this issue is planned for the second phase of the survey.

Another fact shown in the survey data was that the majority of the respondents indicated that the most of their time was spent in detailed design, simulation, and verification from the RTL level down to gates and a final implementation. This result was further reinforced by the data from an essay question in the survey where a majority of respondents felt that the most important application area for VHDL currently is the detailed design of ASICs and FPGAs. Obviously this is currently the case, but we must insure that we gather requirements from the user community that is looking past this into the future where system design at higher levels of abstraction will become a very important application domain for VHDL.

2.2 Language Capabilities for High Level Modeling (Appendix 1 - Charts 11-22)

The main body of the survey was aimed at attempting to determine what additional capabilities the user community desired in VHDL. One area addressed by the survey related to additional support for hardware/software codesign. Additional capabilities in this area were rated as important or very important by a majority of respondents. Interestingly though, the addition of a standard programming language interface (PLI), which would seem to be a requirement in this area and is already underway within the VHDL standardization community, was rated as average importance by a majority of respondents. Further investigation of the user's needs in this area and why perhaps they do not feel that a standard PLI is a part of hardware/software codesign support will be one of the focus areas of the phase II survey.

Another area explored in the survey was that of "system-level" design, or designing at higher levels of abstraction. Within that area, the respondents indicated that the capability to express system-level constraints for power, timing, etc. was very important. On the

other hand, they indicated that an interface to a high level language such as SLDL, which currently is concentrating on the area of constraint specification was only of average importance. This apparent conflict may be the result of the user's desire to express speed, area and power constraints for synthesis within the language itself instead of as separate inputs to the synthesis tools as is now required.

Also in the area of system-level modeling, the respondents indicated that system-level performance and stochastic modeling and better communications modeling capabilities were of average importance but that better abstraction and encapsulation capabilities were of average to high importance. This response is more than likely a result of the respondents current use emphasis of VHDL for the detailed design level than a lack of support for increased modeling capabilities at higher levels of abstraction. This latter area of use is seen as critical to the ability to design next-generation systems by many experts in the HDL area. Therefore, detailed investigations of requirements for VHDL with respect to abstract system-level modeling and design will be a major area of emphasis for the next phase of this process.

The respondents to the survey indicated that high-level synthesis from the behavioral level to the RTL level is important and that is generally the impression one gets from the tool vendors as well. Exactly what requirements this area may generate in the language, if any is unclear, but will be investigated. In addition, a majority of the respondents indicated that the ability to perform fault simulation within VHDL is of relatively low importance. The respondents of the survey were primarily design engineers. The test engineering community was under-represented. This fact could result in some important requirements in the area of production testing being overlooked. To avoid this, we will ensure that the test community, such as those involved in the development and standardization of WAVES, are included in the phase II survey effort.

In the system-level modeling area, the survey asked about support for modeling dynamically reconfigurable hardware and faster simulation via more efficient communications mechanisms, both of which were rated as average importance. Perhaps the dynamically reconfigurable computing area is still too immature to generate much interest and the users may be unaware of the benefits that modeling communications without using VHDL signals would bring. However these might be very fruitful areas of use for exploitation of VHDL and will be examined in more detail.

Finally, the survey results pointed out that a majority of the community seems to be unaware of the currently ongoing work in the areas of Object Oriented VHDL (OOVHDL) and System and Interface Design (SID) and potential future benefit from development of this work. The groups involved should address this lack of awareness and must ensure that the requirements expressed by the users are "filtered" in such a way that the requirements that will be addressed by OOVHDL and SID are not duplicated.

2.3 Language Capabilities for Intellectual Property (IP) Generation and Reuse (Appendix 1 - Charts 23-25)

Several questions in the survey asked about language support for IP generation and reuse. The respondents indicated that capabilities for reuse were of high importance, but what specific requirements exist outside of the areas of encapsulation and abstraction that may be addressed by OOVHDL is unclear. Also, support for IP creation, protection, delivery

and use was rated as high to very important, but specific requirements in this area are unclear as well. Especially since the support of emerging standards in this area such as VSI and OMF were rated as average to low importance. Clearly this is an area where more study is needed.

2.4 Removal of Deprecated Language Features (Appendix 1 - Charts 26-27, 31-32)

When asked which current features in VHDL should be removed to simplify the language, a significant percentage of the respondents indicated "none." In addition, a majority of the respondents indicated that, if a tradeoff in compatibility between VHDL 200x and earlier versions of VHDL vs. increased capability AND ease of use were to be made, compatibility should be the overriding concern. This result clearly indicates that any removal of features from the language should be done carefully and with an eye towards the capability of continued use of legacy code.

2.5 User Interface (Tool) Issues (Appendix 1 - Charts 28-30)

The survey asked a question about the point in the design process where the users spent a majority of their time and many of the respondents indicated that to be simulation/verification. This response seems to be reinforced by the responses to questions on user interface and tool issues. In this area, the users indicated that accelerated simulation via cycle-based, synthesizable or other subset standardization was of high importance. In addition, they indicated that a standard simulation control language is of high importance. Both of these improvements would help reduce the time they need to run the large numbers of simulations (automated via simulation scripts) that are required to verify a design as it is refined to an implementation. The requirement of a standard simulation database format was rated as somewhat less important perhaps indicating that verification data analysis is not currently a problem.

3 The Next Step

The next step in the requirements process is to conduct the second phase of the user survey. This phase will begin by selecting candidates for follow-up interviews. These interviews will be focused on in-depth discussions of the user's needs for the future and how these needs can translate into requirements for VHDL 200x.

Once the second phase of the user survey is completed, the detailed requirements generation process will begin. Detailed requirements for changes or additions to the VHDL language will be identified and documented. These requirements will be reviewed in detail with the VHDL language development community (the IEEE CS VASG and other working groups within the DASC), the VHDL tool vendors, and the Air Force sponsors of this study, to ensure that the recommended requirements have the greatest chance of being addressed by the language update.

The recommended organization of the detailed requirements section of the Phase II document is as follows:

3.1 Existing Language Application Area

This is the section that will describe the new requirements for the language in the area where it is presently applied by a majority of the users, i.e., from the behavioral level down through RTL to gates.

3.1.1 Bug Fixes

Outline any "maintenance" updates required, taking into account those fixes now underway for VHDL 1999 (expected to be renamed as VHDL 2000).

3.1.2 Additional Features

Outline any additional features that are proposed to be added to support the current application area. Examples may be features such as adding scoping rules for signals similar to what is possible in Verilog to help in test bench design, or to add needed features to support the 1164 package such as the addition of reduction operators (urinary AND, OR, etc.) for vector types and the definition of an "image" attribute for all enumerated types.

3.1.3 Deprecated Features

Deprecated features, such as groups, will be considered for removal.

3.2 New Language Application Areas

This section will outline new requirements for the language to extend/better facilitate their use in new application areas, i.e., at higher levels of abstraction. Examples might include:

- Behavioral Synthesis
- SID
- OOVHDL
- Interface to SLDL
- System-level constraint specification

Appendix 1

**Charts presented in the VHDL 200x workshop session
October 1999 VIUF Conference**

**Describes the study
Summarizes the data captured at that time**
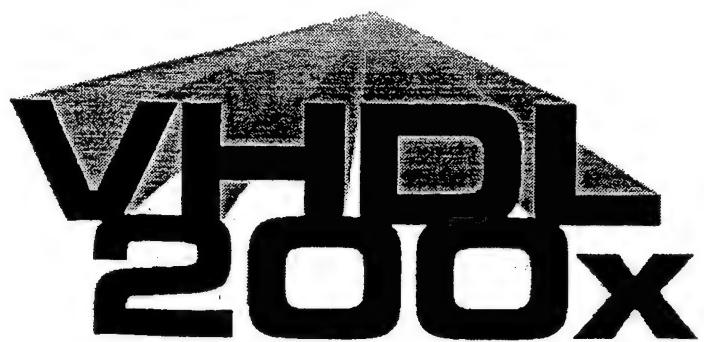
VHDL 200x

User Survey Results

James Aylor - University of Virginia
Robert Klenke - Virginia Commonwealth University

Air Force VHDL 200x Project

- **Funded by Avionics Directorate of Wright Lab**
- **Project goals:**
  - ◊ **Provide assistance to the community in the development of the next version of VHDL**
  - ◊ **Insure that the needs of the Air Force (and others) are duly considered**
- **Program duration of Sept. 1998 - Nov. 2001**
- **Team members:**
  - ◊ **University of Virginia**
  - ◊ **Virginia Commonwealth University**
  - ◊ **Advanced Technology Institute**
  - ◊ **Menchini & Associates**
  - ◊ **EDA Standards Consulting**

Project Deliverables

- VHDL200x Requirements Document
  ◊ Establish requirements based on work of VASG, needs of the Air Force, and team's experience
- VHDL 200x Language Extensions Document
  ◊ Define adequate extensions to address requirements that will be supported by implementations
- VHDL200x Language Reference Manual
  ◊ Provide assistance in developing LRM for update

Proposed Requirements Document Development Process

Page 2

9

2

Proposed Requirements Document Development Process

VHDL 200x — University of Virginia/Virginia Commonwealth University — 5

Demographics

- **Which HDL do you use?**
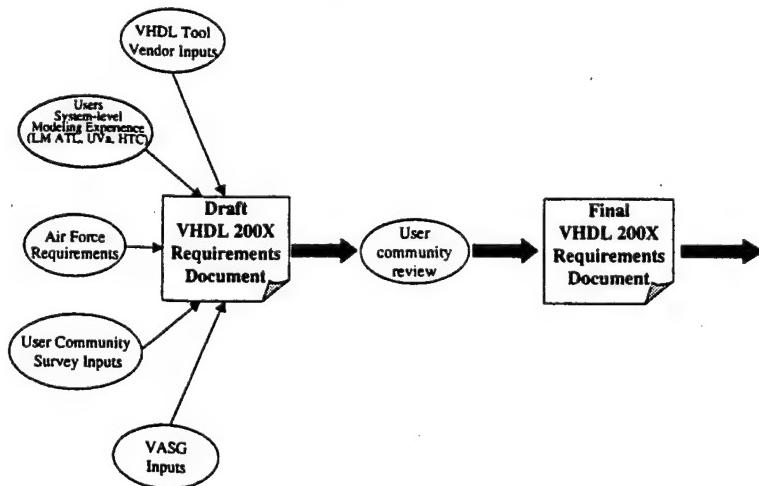
VHDL 200x — University of Virginia/Virginia Commonwealth University — 6
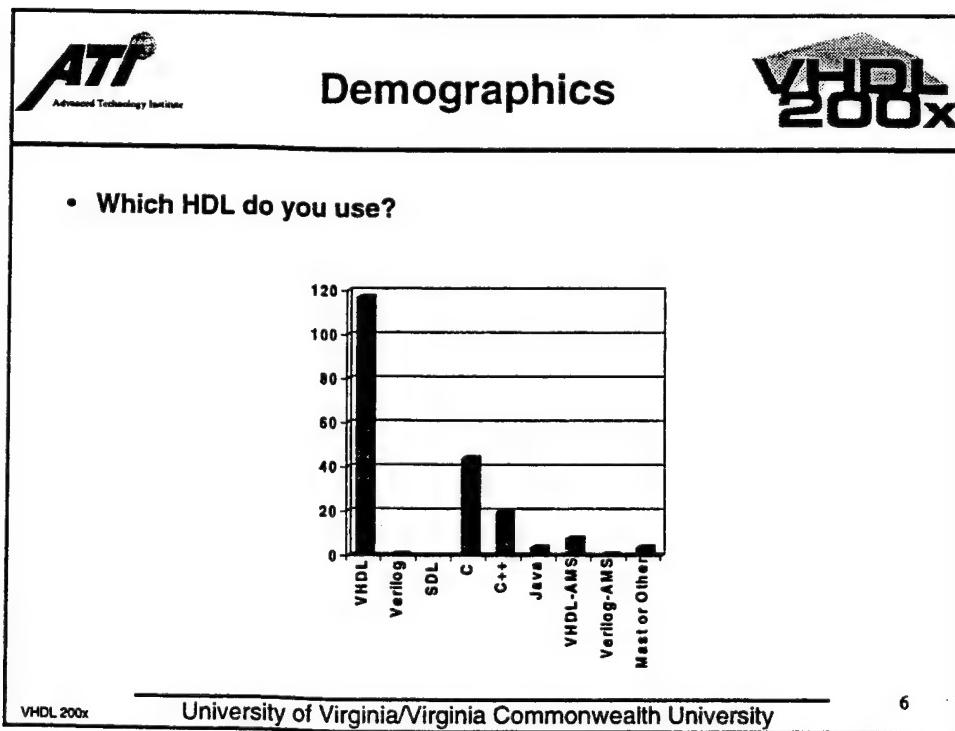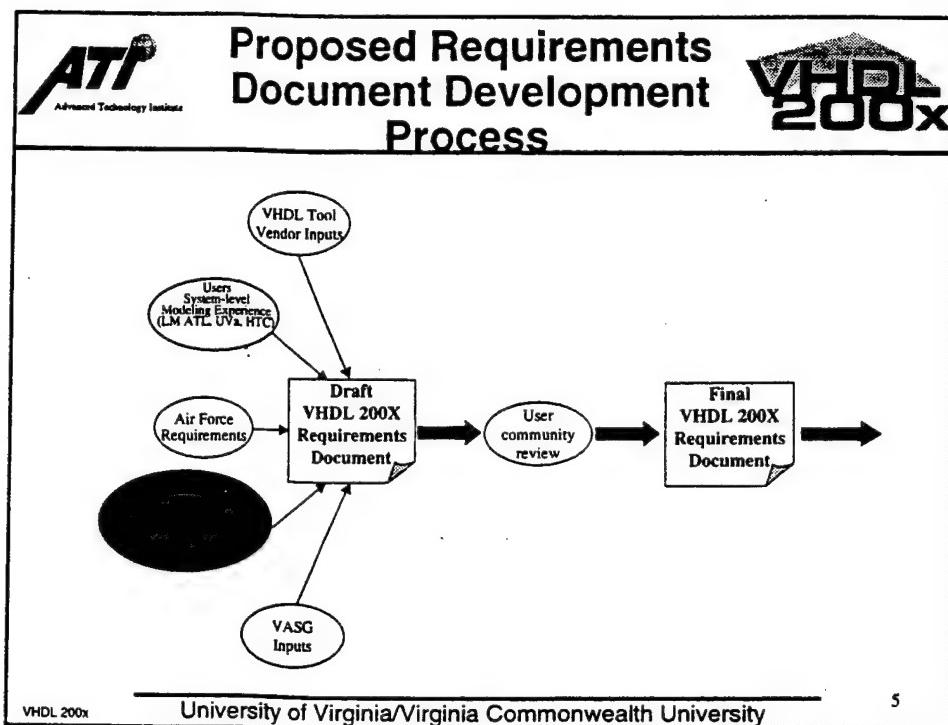
Demographics (cont.)

- VHDL is used in my organization for?
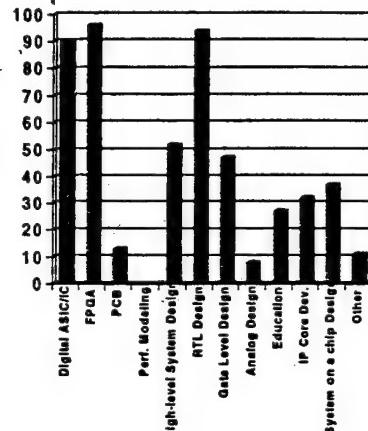
University of Virginia/Virginia Commonwealth University

7

Demographics (cont.)

- Primary end application of my designs are?
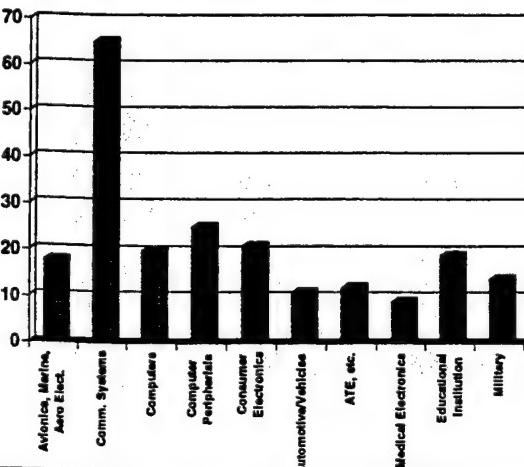
University of Virginia/Virginia Commonwealth University

8

Page 4

11

4

Demographics (cont.)

- Where in the design cycle is the majority of time spent in your organization?
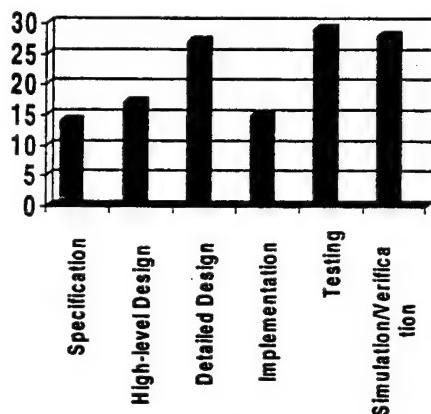
Demographics (cont.)

- What is the current most important application area of VHDL?

ATI — Advanced Technology Institute

Language Capabilities
High Level Modeling

VHDL 200x

- **Hardware/Software co-design support ?**

ATI — Advanced Technology Institute

Language Capabilities
High Level Modeling (cont.)

VHDL 200x

- **Standard programming language interface by which externally generated (foreign, e.g. C, C++) models and functions can be included in the simulation of a design ?**

Language Capabilities
High Level Modeling (cont.)

- System-level constraint specification within VHDL (power, timing constraints, etc.) ?

Language Capabilities
High Level Modeling (cont.)

- System level modeling (e.g. performance and stochastic modeling) ?

Language Capabilities
High Level Modeling (cont.)

- **Better encapsulation and abstraction capabilities ?**
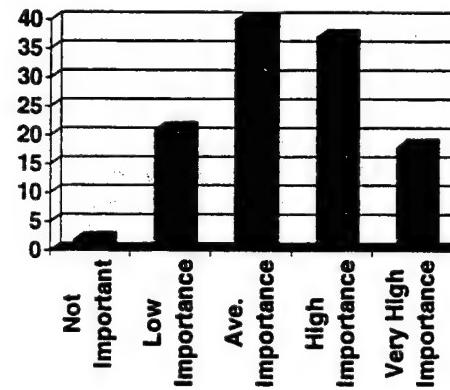
VHDL 200x · University of Virginia/Virginia Commonwealth University · 15

Language Capabilities
High Level Modeling (cont.)

- **Better communications modeling capabilities ?**
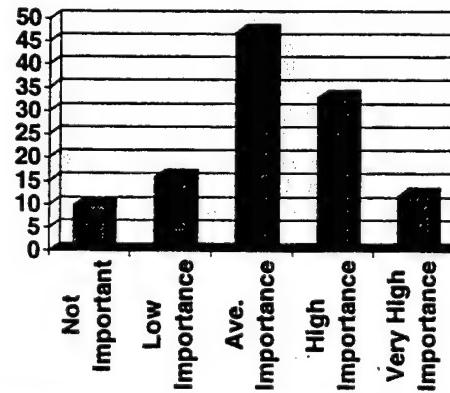
VHDL 200x · University of Virginia/Virginia Commonwealth University · 16

Page 8

8

- Ability to interface to high-level system description languages (SDL, SLDL) ?

![ATI Advanced Technology Institute logo]
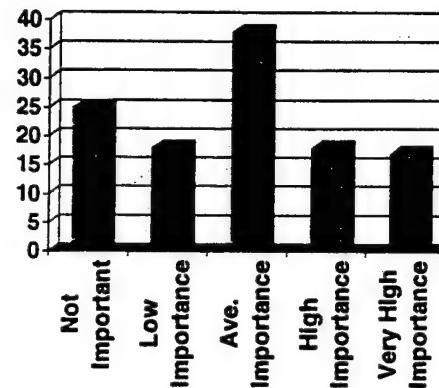
Language Capabilities
High Level Modeling (cont.)

- Are you aware of the following extensions to VHDL currently being pursued and do you feel they will be useful in addressing some of the capabilities described above ?

Legend:
- ■ OOVHDL
- ■ SID

Page 9
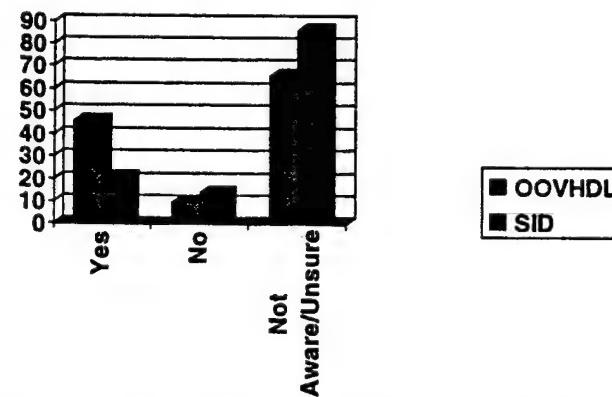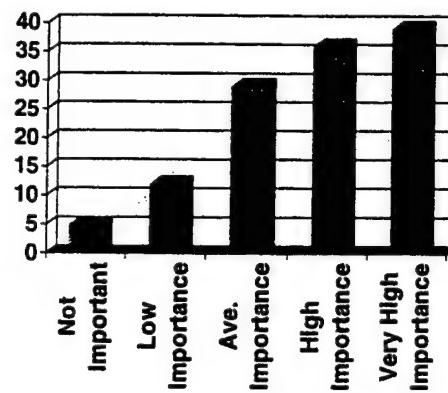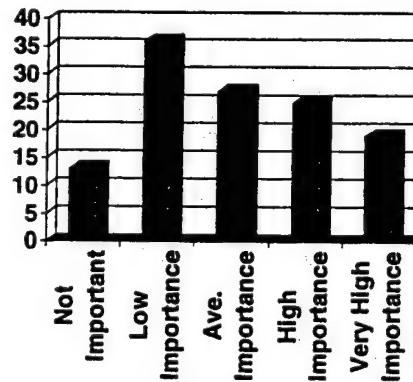
Language Capabilities
High Level Modeling (cont.)

- Support for new hardware design paradigms such as data flow and reconfigurable hardware ?
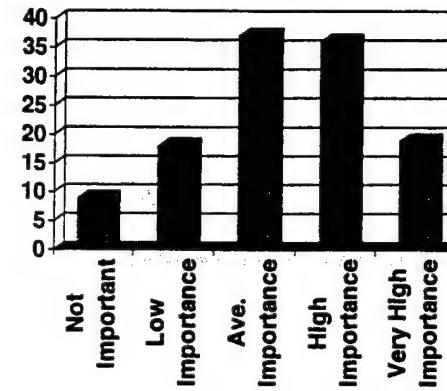
VHDL 200x   University of Virginia/Virginia Commonwealth University   21

Language Capabilities
High Level Modeling (cont.)

- Accelerated simulation via more efficient interprocess communication mechanism (messages, shared variables, lightweight signals, etc.)?

VHDL 200x   University of Virginia/Virginia Commonwealth University   22
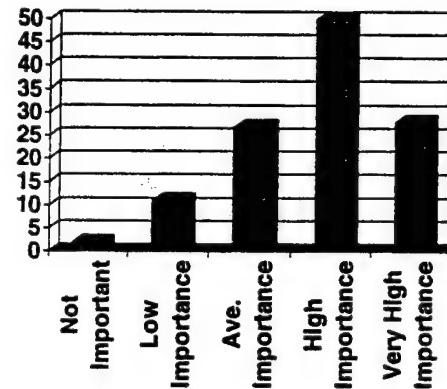
Page 11

18

1

ATT
Advanced Technology Institute

Language Capabilities
Reuse/IP

VHDL
200x

- **Better reuse capabilities ?**

ATT
Advanced Technology Institute

Language Capabilities
Reuse/IP (cont.)

VHDL
200x

- **Support for externally generated models that conform to OMF and VSI emerging standards ?**

Page 12

19

Language Capabilities
Reuse/IP (cont.)

- Support for IP creation, protection, delivery, and reuse ?
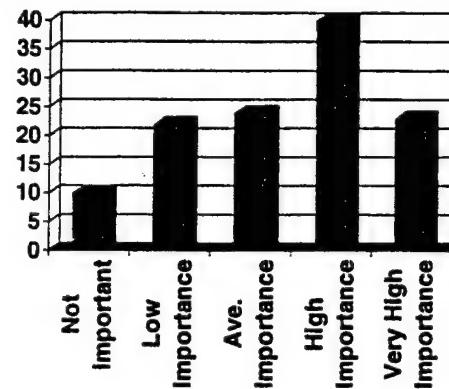
University of Virginia/Virginia Commonwealth University

VHDL 200x   25

Language Capabilities
Unused Features

- In the interest of simplifying VHDL what language constructs do you feel could be eliminated?

University of Virginia/Virginia Commonwealth University

VHDL 200x   26

Page 13

20
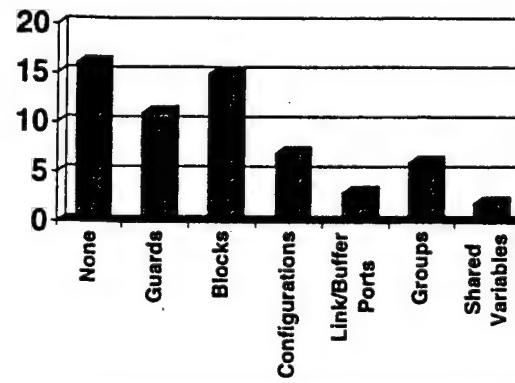
Language Capabilities
Unused Features - Previous Data

Q. 24. Features being currently used and planning to use
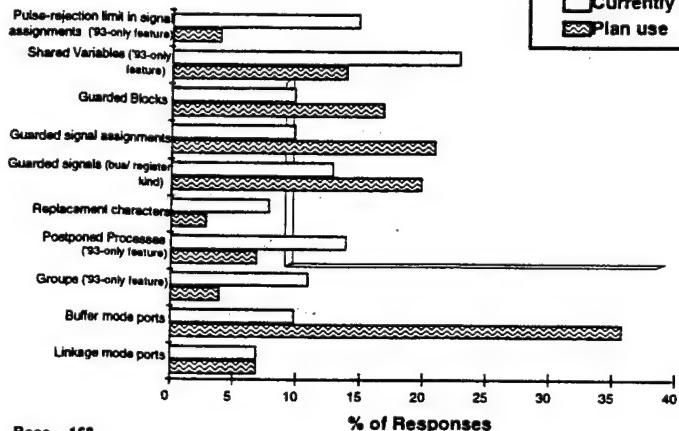
Key:
- Currently use
- Plan use

Features (top to bottom):
- Pulse-rejection limit in signal assignments ('93-only feature)
- Shared Variables ('93-only feature)
- Guarded Blocks
- Guarded signal assignments
- Guarded signals (bus/ register kind)
- Replacement characters
- Postponed Processes ('93-only feature)
- Groups ('93-only feature)
- Buffer mode ports
- Linkage mode ports

X-axis: 0, 5, 10, 15, 20, 25, 30, 35, 40 — **% of Responses**

Base = 168

Language Capabilities
User Interface Issues

- **A standard simulation control language ?**
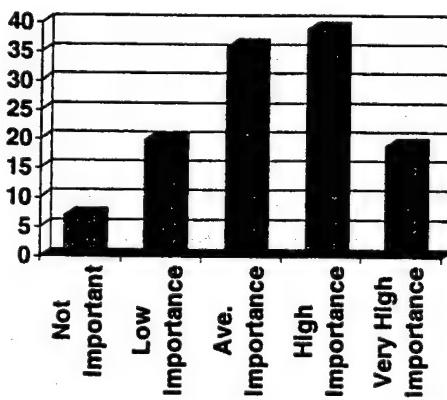
Y-axis: 0, 5, 10, 15, 20, 25, 30, 35, 40

Categories: Not Important, Low Importance, Ave. Importance, High Importance, Very High Importance

Language Capabilities
User Interface Issues (cont.)

- **A standard simulation waveform database format ?**
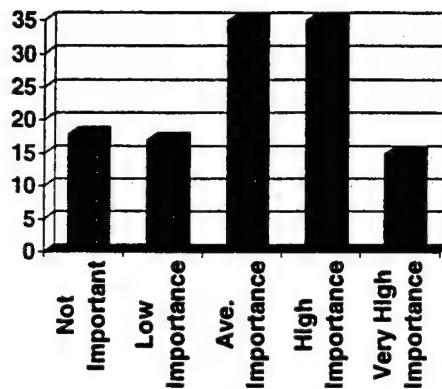
Language Capabilities
User Interface Issues (cont.)

- **Accelerated simulation via cycle-based, synthesizable or other subset standardization ?**

Language Standardization

- How important is it that the next update of VHDL be backwards compatible with previous versions of VHDL ?

Language Standardization
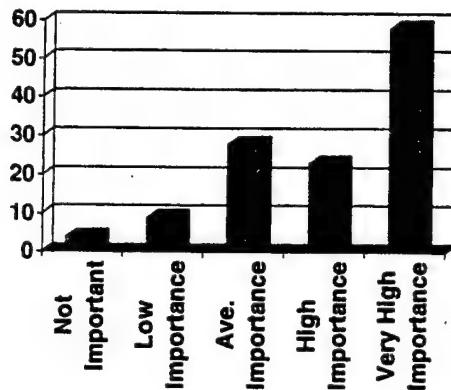
- In designing the next update to VHDL, if a choice between language capability and readability/ease of use vs. backward compatibility has to be made for one or more constructs, would you prefer strict backward compatibility or less capability in the revised language
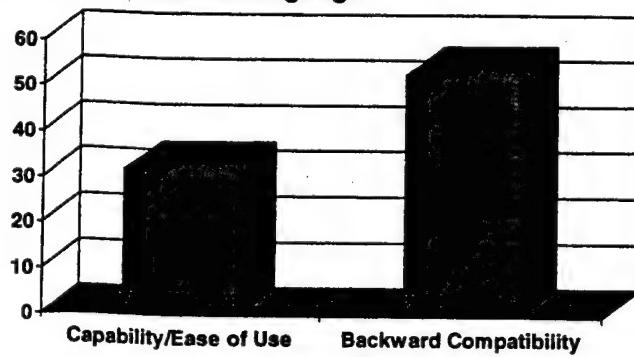
Conclusions

- Demographics
  - ◊ Detailed design and verification is still the primary application
  - ◊ VHDL for synthesis continues to be important
  - ◊ C and C++ seem to be gaining popularity for modeling hardware

Conclusions

- High Level Modeling
  - ◊ High level synthesis is in great demand
  - ◊ Codesign remains a critical need,
  - ◊ but a standard PLI doesn't seem to be a major concern
  - ◊ System-level constraint specification seems important (driven by the desire to specify timing for synthesis in the language, not the tool?)
  - ◊ System-level modeling capabilities are still not a first-level concern for mainstream users
  - ◊ The community seems to be relatively unaware of existing efforts in this area (SID, OOVHDL, SLDL)
  - ◊ Production testing (fault simulation) appears to not be a critical issue for designers

Conclusions

- **Reuse/IP**
  - ◊ Support for industry standard reuse formats is not high,
  - ◊ but reuse itself seems to be important (perhaps everyone wants to be able to reuse their own designs?)
- **Unused Features**
  - ◊ A significant portion of the users don't want to remove any features,
  - ◊ but nobody seems to use guards, blocks, or groups
- **Standardization**
  - ◊ Backward compatability is a must

Group Contacts

- **Jim Aylor, University of Virginia**
  jha@virginia.edu
- **Bob Klenke, Virginia Commonwealth University**
  rhklenke@vcu.edu
- **Paul Menchini, Menchini & Associates**
  mench@mench.com
- **Ronald Waxman, EDA Consulting**
  r.waxman@computer.org
- **Jack Stinson, Advanced Technology Institute**
  stinson@scra.org
- **Bill Anderson, Advanced Technology Institute**
  Anderson@aticorp.org

Proposed Requirements Document Outline

- **Existing Language Application Area**
  - ◊ **Bug Fixes**
  - ◊ **Additional Features (scoping, 1164 package)**
  - ◊ **Removal of Unused Features**
- **New Language Application Areas**
  - ◊ **SID**
  - ◊ **OOVHDL**
  - ◊ **Interface to SLDL**
  - ◊ **System-level constraint specification**
  - ◊ **...**

VHDL 200x   University of Virginia/Virginia Commonwealth University   37

Appendix 2

Compiled raw data captured in Phase I of study

1. Which HDL do you use?	
VHDL	121
Verilog	3
SDL	0
C	46
C++	21
Java	4
VHDL-AMS	0
VERILOG-AMS	1
MAST or other Analog language	4
Other	28
Written Response for Question 1 under Other:	
Altera HDL	
Matlab	
Modelica	
DSL,ABEL	
SIMULINK	
VHDL+	
Perl	
Pspice	
Module Compiler	
SpectreHDL	
Handel-C	
AHDL ALTERA	

2. VHDL is used in my organization for?	
Digital ASIC/IC	93
FPGA design	99
PCB	13
Performance Modeling	2
High-Level System Design	53
RTL Design	98
Gate Level Design	48
Analog design	8
Education	30
IP core development	35
System-on-a-chip design	38
Other	11
Written Response for Other:	
Nothing	
System Verification	
Simulation models	
MEMS/Microsystem modeling	
Non-electronics simulation (rarely)	
we've developed a PD analyzer for simulation	

1

3. VHDL is primarily used in my organization for?	
Digital ASIC/IC design	43
FPGA design	45
PCB	0
Performance Modeling	2
High-Level System Design	1
RTL Design	18
Gate Level Design Analog design	0
Education	9
IP core development	2
System-on-a-chip design	5
Other	5
Written Response for Other:	
we develop VHDL and Verilog compilers	
Digital ASIC/IC simulation	
MEMS/Microsystem modeling	

4. I personally use VHDL for?	
Digital ASIC/IC design	73
FPGA design	74
PCB	3
Performance Modeling	21
High-Level System Design	39
RTL Design	75
Gate Level Design	23
Analog design	5
Education	0
IP core development	20
System-on-a-chip design	0
Other	16
Written Response for Other:	
switched to verilog	
writing test examples in VHDL for testing my compiler	
Test samples and component modelling	
simulation environments	
MEMS/Microsystem modeling	

2

5. My primary job responsibility is?	
Corporate management	0
Design engineering	0
Design engineering management	0
Engineering/technical support	0
Research and development	0
Technical marketing	0
EDA tool development	0
IP core development	53
Other	7
Question NOT Answered by	72
Written Response for Other:	
Professor (Engineering)	
Academic	
HDL/Design Reuse Consulting	
Consulting	
Design verification	
Teaching	
staff technologist	

6. Primary end application of my designs are?	
Avionics Marine Aerospace Electronics	19
Communication systems	67
Computers	22
Computer peripherals	27
Consumer electronics	22
Automotive/Other Ground vehicles	12
Industrial Controls Robotics	2
Electronic Instrumentation/ATE/Design and Test	13
Medical Electronics Appliances	10
Educational institution	20
Military	15
Other	20
Written Response for Other:	
DSP	
none, purely research for its own sake	
Government	
Dynamically reconfigurable logic applications	
Academic Research	
outsourcing/consultancy	
outsourcing/consultancy	
EDA tools	
VME-bus DSP boards	
Video and image processing	

3

7. Please estimate the relative percentage use of each capture method for your VHDL designs ?	
Text Editor - 0%	1
Text Editor - 10%	10
Text Editor - 20%	13
Text Editor - 30%	2
Text Editor - 40%	6
Text Editor - 50%	10
Text Editor - 60%	2
Text Editor - 70%	6
Text Editor - 80%	10
Text Editor - 90%	30
Text Editor - 100%	36
Language-sensitive Editor - 0%	43
Language-sensitive Editor - 10%	8
Language-sensitive Editor - 20%	3
Language-sensitive Editor - 30%	4
Language-sensitive Editor - 40%	2
Language-sensitive Editor- 50%	6
Language-sensitive Editor - 60%	7
Language-sensitive Editor - 70%	11
Language-sensitive Editor - 80%	11
Language-sensitive Editor - 90%	11
Language-sensitive Editor- 100%	20
Schematic Editor - 0%	95
Schematic Editor - 10%	15
Schematic Editor - 20%	8
Schematic Editor - 30%	2
Schematic Editor - 40%	2
Schematic Editor- 50%	4
Schematic Editor - 60%	0
Schematic Editor - 70%	0
Schematic Editor - 80%	0
Schematic Editor- 90%	0
Schematic Editor- 100%	0
State machine/chart/table Editor - 0%	99
State machine/chart/table Editor - 10%	18
State machine/chart/table Editor - 20%	6
State machine/chart/table Editor - 30%	1
State machine/chart/table Editor - 40%	0
State machine/chart/table Editor- 50%	2
State machine/chart/table Editor - 60%	0
State machine/chart/table Editor - 70%	0

4

State machine/chart/table Editor - **80%**	0
State machine/chart/table Editor- **90%**	0
State machine/chart/table Editor- **100%**	0
Truth Table Editor - **0%**	122
Truth Table Editor - **10%**	4
Truth Table Editor - **20%**	0
Truth Table Editor - **30%**	0
Truth Table Editor - **40%**	0
Truth Table Editor- **50%**	0
Truth Table Editor - **60%**	0
Truth Table Editor - **70%**	0
Truth Table Editor - **80%**	0
Truth Table Editor- **90%**	0
Truth Table Editor- **100%**	0
Reused design models - **0%**	72
Reused design models - **10%**	25
Reused design models - **20%**	12
Reused design models - **30%**	5
Reused design models - **40%**	4
Reused design models - **50%**	7
Reused design models - **60%**	0
Reused design models - **70%**	0
Reused design models - **80%**	1
Reused design models - **90%**	0
Reused design models - **100%**	0
Other - **0%**	123
Other - **10%**	1
Other - **20%**	1
Other - **30%**	0
Other - **40%**	1
Other - **50%**	0
Other - **60%**	0
Other - **70%**	0
Other - **80%**	0
Other - **90%**	0
Other - **100%**	0

8. If you are using VHDL synthesis, please estimate the percentage of your designs that are synthesized to each of the following?	
VITAL gates - **0%**	66
VITAL gates - **10%**	8
VITAL gates - **20%**	7
VITAL gates - **30%**	4
VITAL gates - **40%**	1
VITAL gates - **50%**	9
VITAL gates - **60%**	1
VITAL gates - **70%**	6
VITAL gates - **80%**	2
VITAL gates - **90%**	2
VITAL gates - **100%**	20
Other VHDL gate-level descriptions - **0%**	81
Other VHDL gate-level descriptions - **10%**	4
Other VHDL gate-level descriptions - **20%**	4
Other VHDL gate-level descriptions - **30%**	4
Other VHDL gate-level descriptions - **40%**	1
Other VHDL gate-level descriptions - **50%**	3
Other VHDL gate-level descriptions - **60%**	3
Other VHDL gate-level descriptionss - **70%**	4
Other VHDL gate-level descriptions - **80%**	3
Other VHDL gate-level descriptions - **90%**	3
Other VHDL gate-level descriptions - **100%**	16
Verilog gates - **0%**	96
Verilog gates - **10%**	5
Verilog gates - **20%**	1
Verilog gates - **30%**	4
Verilog gates - **40%**	0
Verilog gates - **50%**	5
Verilog gates - **60%**	2
Verilog gates - **70%**	0
Verilog gates - **80%**	3
Verilog gates - **90%**	3
Verilog gates - **100%**	7
Other non-VHDL or Verilog gate level descriptions - **0%**	71
Other non-VHDL or Verilog gate level descriptions - **10%**	2
Other non-VHDL or Verilog gate level descriptions - **20%**	1
Other non-VHDL or Verilog gate level descriptions - **30%**	2
Other non-VHDL or Verilog gate level descriptions - **40%**	1
Other non-VHDL or Verilog gate level descriptions - **50%**	3
Other non-VHDL or Verilog gate level descriptions - **60%**	0
Other non-VHDL or Verilog gate level descriptions - **70%**	1

6

Other non-VHDL or Verilog gate level descriptions - **80%**	1
Other non-VHDL or Verilog gate level descriptions - **90%**	3
Other non-VHDL or Verilog gate level descriptions - **100%**	20

9. Hardware/Software co-design support ?	
not important	7
low importance	14
average importance	24
high importance	54
very high importance	25

10.System-level constraint specification within VHDL (power, timing constraints, etc.) ?	
not important	7
low importance	25
average importance	24
high importance	36
very high importance	34

11. Fault simulation with VHDL ?	
not important	16
low importance	36
average importance	28
high importance	25
very high importance	19

12. System level modeling (e.g. performance and stochastic modeling) ?	
not important	2
low importance	24
average importance	45
high importance	37
very high importance	16

13. High level synthesis (from behavioral models to RTL) ?	
not important	5
low importance	12
average importance	31
high importance	37
very high importance	40

14. Support for new hardware design paradigms such as data flow and reconfigurable hardware ?	
not important	9
low importance	18
average importance	39
high importance	37
very high importance	20

15. Accelerated simulation via cycle-based, synthesizable or other subset standardization ?

not important	8
low importance	14
average importance	32
high importance	41
very high importance	29

16. Accelerated simulation via more efficient interprocess communication mechanism (messages, shared variables, lightweight signals, etc.)?

not important	9
low importance	27
average importance	30
high importance	36
very high importance	22

17. Better encapsulation and abstraction capabilities ?

not important	2
low importance	22
average importance	42
high importance	38
very high importance	18

18. Better reuse capabilities ?

not important	3
low importance	11
average importance	29
high importance	51
very high importance	28

19. Better communications modeling capabilities ?

not important	10
low importance	16
average importance	49
high importance	35
very high importance	12

8

20. Standard programming language interface by which externally generated (foreign, e.g. C, C++) models and functions can be included in the simulation of a design ?	
not important	11
low importance	16
average importance	47
high importance	34
very high importance	14

21. Support for externally generated models that conform to OMF and VSI emerging standards ?	
not important	22
low importance	30
average importance	39
high importance	21
very high importance	7

22. Support for IP creation, protection, delivery, and reuse ?	
not important	10
low importance	22
average importance	25
high importance	43
very high importance	23

23. A standard simulation control language ?	
not important	8
low importance	21
average importance	36
high importance	39
very high importance	21

24. A standard simulation waveform database format ?	
not important	19
low importance	18
average importance	35
high importance	36
very high importance	16

25. Ability to interface to high-level system description languages (SDL, SLDL) ?	
not important	25
low importance	18
average importance	38
high importance	19
very high importance	20

9

26. How important is it that the next update of VHDL be backwards compatible with previous versions of VHDL ?	
not important	4
low importance	9
average importance	29
high importance	23
very high importance	60

27. Are you aware of the following extensions toVHDL currently being pursued and do you feel they will be useful in addressing some of the capabilities described above ?	
OOVHDL	
YES	46
NO	11
Not aware/unsure	69
SID (VHDL+)	
YES	21
NO	14
Not aware/unsure	90

28. Do you feel the extensions to VHDL currently being pursued will be useful to YOU in the future?	
OOVHDL	
YES	48
NO	12
Not aware/unsure	66
SID (VHDL+)	
YES	16
NO	9
Not aware/unsure	101

29. Where in the design cycle (specification, high-level design, detailed design, implementation, testing, etc.) is the majority of time spent in our organization ?

Use of VHDL : detailed design, + lower design phases, Spec/High-level : more via C-models,.. distribution in effort about 50/50.

Detailed design.

detailed design, implementation. Which is actually not good.

simulation and test

High-level design

Implementation & testing

Don't Know

SPECIFICATION

specification and testing about 80 percent of development time

specification - 20% high-level design - 10% detailed design - 20% implementation - 10% testing and re-design - 30% creating layout for ASIC - 10%

detailed design and simulation

design and test

Specification and testing.

Detailed design/implementation

1.   testing 2. high-level design 3. detailed design 4.implementation 5.specification

· specification and testing

Testing, perfomance verification.

testing

Specification

Detail design and behavioral modeling.

high-level design, testing, system integration with SW

The time spent all five cycles mentioned is, dependent on the project, more or less equal.

Spec. High-level design, detailed design and partly implementation

Testing

VHDL simulation and gate-level simulation

Most of the time is spent in debug of the FPGAs/boards

11

Usually in the verification of behavioral and synthesized models.

spec 15% HL design 15% detailed design 35% simulation 25% test/debug 10%

Functional verification

Testing. Simulation and validation of functionality.

High level design / detailed design about equally.

high-level design, detailed design

High-level design

As an educational institution, the majority of teaching work uses RTL VHDL. We are, however, engaged in research work and EDA tool development that uses high-level behavioural VHDL and VHDL-AMS. In that sense, most time is spent on high-level design

design implementation and simulation

development of MEMS/Microsystem device models (behavioral level)

Most of the time is spent on implementation and simulation. About the same amount of manpower on each.

Most of the time is spent during the synthesis and timing spin that occurs to meet timing requirements.

high level design and RTL synthesis

Students tend to be mostly occupied with detail design, but when acting as a consultant I spend most of the time on specification and verification: implementation is not usually a big problem

detailed design

detailed design and testing (simulation)

testing & debug

Not enough is spent on high-level design, it's a lot of detailed dsn and implementation.

Detailed design

detailed design and implementation.

Detailed design and simulation

Detailed design, implementation and testing.

detailed design

detailed design

Design verification

verification.

spec., testing

Spec: 30 % Coding: 15 % Test: 55 %

30. What is the current most important application area of VHDL ?

RTL-design (so detailed design phase)

I still suppose that digital design and FPGAs are the most important application area.

System-level-design using VHDL-AMS FPGA design using VHDL

Education (Design methodology, ASIC and FPGA design)

consumer electronics

As an educational user of VHDL we don't have one

Communications ASIC

System Level design

RTL design

ASIC design

Digital ASIC and FPGA design

RTL design and simulation

Detailed design.

Rapid hardware design, easily migratable across technologies

synthesis simulation

RTL design, system modelling

FPGA design.

system simulation

ASIC and FPGA Design

Synthesizable VHDL (including high level synthesis)

FPGA

FPGA programming and simulation - code maintenance.

Digital design using synthesis tools

In our organization, the emphasis lies on synthesis of RTL level designs. Other applications (system design, board level testing) are upcoming.

13

ASIC design

RTL synthesis

FPGA design.

Modelisation

FPGA design

For me personally, 3G mobile comms

ASIC design.

Complete top down design, from abstract system to VITAL gates.

high-level design, synthesis

embedded systems

RTL synthesis to FPGAs. I forsee this changing in the near future to behavioural synthesis.

design, testbenches and simulation models

Design space exploration on architecture level

From my point of view, FPGA design and simulation; the rest of the industry may disagree!

logic synthesis

ASIC design

Digital ASIC design

communications, telecom

Integrated circuit simulation modeling

ASIC design

FPGA design entry

Communication equipment

design education

high-level design

Design of FPGA's

RTL entry and the verification of that design. Behavior level is becoming more important as we understand the subset that is useful in high performance asics and micro processors

high-level, FPGA-design, prototyping

14

Synthesis to Gates

31. Are there any other areas where you use VHDL that were not covered by questions 2-4 ?

Yes E.g. making an VHDL-model of analog modules so that a single simulator environment is possible (using "reals") (translation from HDLA)

NO.

No

No

no

no

No, far as I can think of

no

no

no

no

Test vector generation generated from the VHDL testbench

no

Test benches/ System verification

no

no

no

no

no

no

Testbench creation. Specifically, bus-functional-models and bus-checkers.

no

embedded systems

no

no

no

no

Simulation behavioral models

high-level testbenches

no

Yes, Abstract system modelling. For example, I have used VHDL to model air interfaces and such for communication channels.

no

Writing of test benches.

no

no

automatic control others ?

32. In designing the next update to VHDL, if a choice between language capability and readability/ease of use vs. backward compatibility has to be made for one or more constructs, would you prefer strict backward compatibility or less capability in the revised language?

backwards compatibility is a must, else why not switch to other languages like C/C++ and extend them with some signal typesC/C+

Backward compatibility.

less capability. The language is already much to complicate for conformance for average VHDL-AMS simulators. Keep the basics straight. Skip the not used constructs.

I would prefer to improve language capability and ease of rather than backward capability

It is very important the backward compability.

I am prepared to accept some loss of backwards capability if it can be justified

Ease of use is always good, but depends to what extent the old code has to be rewritten.

I would prefer strict backward compatibility.

more capability and ease of use

strict backward compatibility

I would definitely prefer language capability

depends on the number of constructs and on the impact on reuse of modules

strict backward compatibility.

16

Strict backward compability.

depends too much on the suggested additional capabilities vs compatibility trade-off to be able to answer. Cf. files in VHDL87->93: incompatible but easily adaptable to new standard.

Some loss of backward compatibility would be acceptable.

less capability

backward compatibility

Yes? strick backward compatibility over more capability.

Prefer more language (and library) simplification and consistency and more object-oriented capability.

BAckward compatibility is important. Incompatibilities and upgrade guidelines have to be well documented

In my opinion, a clean design of VHDL is necessary. Therefore backward compatibility should be preserved wherever possible, but readability and ease of use should be given higher precedence.

No, I would prefer more language capability.

Dtich the strict backward compatibility and go for the extra features. C to C++ broke a few things, but they were so minor that no one complained. Heck, right now we spend a lot of time whining about how we have to take perfectly good VHDL '93 and turn into back into VHDL '87 to get FPGA Express to synthesize it, I doubt we'll complain any more if we have to change some VHDL '93 into VHDL '2000.

I prefer a strict backward compatibility in the next update. A further update could include more language capability. I think that improvement of language capability would be done step by step.

So long as the synthesizable subset of VHDL is backwards compatible, it doesn't really matter.

Go for the improved capability, particularly if there is a clear mapping for VHDL'93 to VHDL-2000 in those areas. The old compilers will still be there for a while, and most products will have a command-line option for which syntax to use anyway.

Don't you mean *more* capability in the revised language? I believe that extra capability, and language extensions, should take precedence over backward compatability. If you stick with backward compatibility as a primary requirement, then VHDL will soon be obsoleted by the next generation of HDLs.

I would say that language capability/readability/ease of use has priority over backward compatibility.

I'd prefer readability/ease of use over backward compatibility. Re-use is very low, however, breaking all of the existing tools must be considered as very undesirable.

Strict backwards compatibility is very important. The language as currently defined is quite capable of doing most of the things asked of it. Complicating it runs the risk of making it less usable for the majority of practitioners. Change for change sake is always a bad idea.

Prefer backward compatibility, with the exception of minor incompatibilities, such as new reserved words and relaxation of restrictions.

less capability

I would be prepared to sacrifice backward compatability for better ease of use.

I would be willing to trade off some backward compatibility for enhanced capability.

I prefer more capability, ease of use, readability. I could tolerate a loss of backward capabiltiy.

Less capability

Backward compatibility is extremely important, but it can be sacrificed for better language capability **if** new compilers can clearly identify the incompatibility and there is a clear remediation path

strict backward compatibility

Backward compatibility

If the capability is unneeded and it simplifies the language, the by all means scrap it!

I prefer capability and ease-of-use over bkwd-compat. (can always run a tool in a bkward-compat. mode).

I would prefer strict backward compatibility, but optimized performance on a well-defined subset with less capability.

Less capability (but not to throw away the entire language)

Backward compatibility less important than significant improvement in language

I would prefer increased capability and ease of use to backward compatibility.

backward compatibility can be dropped for less frequently used features

New versions should be 100% backward compatible, new features can be added

My preferance is upward compatibility.

I really don't know. The maintainance and reuse of "old" models need backward capability.

Ease of Use is very important. Backward compatible to an RTL Subset of the language is very important. Backward compatible otherwise some importance (see comment in annoying features). As a VHDL trainer, I have found that hardware designers are just that - hardware designers. Please try to keep the usage of VHDL for hardware design simple. I just saw that one simulator vendor thinks that with VHDL-93 that configurations are required for all code. Configurations are a foreign to most hardware designers.

33. What design languages (including any from question 1 including VHDL and SLDL) do you plan to use in the next 2 years, for what purpose, and why the specific language?

C/C++ : making behavioral models VHDL : rtl/detailed design Verilog : only for gatelevel simulation (since best interface to backend tools) (Place&Route tools do not like VHDL gatelevel netlists)

VHDL-AMS. FOR mixed simulation and modeling. Because it has become a standard.

VHDL-AMS (we are using already) since there is no other non propriatary standard language for this purpose

VHDL (FPGA and ASIC design) VHDL-AMS (mixed-mode system design, MEMs modeling)

18

45

Unknown.

VHDL - I need as high a level of description as I can find (in terms of compiler syntax, rather than abstraction level). This is why I don't use Verilog. Purpose: primarily FPGA and IP synthesis and simulation. C – for everything else.

I only foresee using VHDL in the next 2 years.

Verilog, for existing IP that I must incorporate in my design. Module Compiler, for datapath design in communications ASICS.

VHDL - continuing same purposes VHDL-AMS - same purposes as VHDL SLDL (Rosetta) - language design and validation

VHDL, SLDL, MSC, C, C++, (OOVHDL) for system verification design languages used in embedded system designs

VHDL - behavioural synthesis C (subset) - behavioural synthesis VHDL-AMS - development of EDA tools

vhdl in much the same way as we are using it today because we are comfortable and efficient with it

SpectreHDL: because it is the only one which is implemented fully functional. VHDL-AMS: if I get access to a simulator that provides a fully functional language implementation, including tolerances and timestep control.

VHDL will be used for all our ASIC logic entry.

VHDL for behaviour and RT level, C and/or SDL for system level design, incl co-design

VHDL because the rest of the world expects it for FPGA Verilog because most of California expects it for FPGA and ASIC Handel-C because I fervently believe in communicating- sequential-processes as a design paradigm

vhdl only

VHDL only for ASIC design. Experience base, tools support

VHDL, because that's what I know and that's what the company uses.

VHDL, SLDL, possibly C/C++ (which may be part of SLDL env.)

C for high-level procedural models. C allows for quick turn around time for design changes. VHDL for RTL models and for structural representations of designs. The formality of VHDL is an asset when working in large teams concurrently developing IC models.

VHDL, Verilog, Vera (Maybe), Sometimes TCL and Perl also come in handy.

VHDL, Modelsim macrofiles, UNIX script (awk) I use awk-scripts to generate port maps etc. Macrofiles are used to control the simulation. VHDL is used for testbench and RTL design.

VHDL For FPGA Design It is widely used and has proved efficient for FPGA designs

VHDL Esterel for codesign

VHDL

VHDL

VHDL primarily. We'll have a look at SLDL and VHDL+, but my present view is that VHDL can handle all the aspects of system design that we are interested in.

VHDL, Verilog: High-level + RTL models C/C++: Spec + High-level models; HW/SW-designs

34. In the interest of simplifying VHDL what language constructs do you feel could be eliminated?

difference output and buffer types (make outputs also readable), would definitely fix lots of problems in gatelevels (and avoid use of massive unnecessary signals)

Not sure.

This is the wrong question. We have made a proposal for VHDL-AMS subset (which is similar to 1076.6), this is pretty much sufficient to me.

Guarded signals LINKAGE and BUFFER ports

blocks

None

I lost track after working at CLSI :-)

guarded assignements linkage selected signal assignements conditional signal assignements

architecture configuration passive processes inside the entity declaration

groups components

Capture a schematic of a small, simple FPGA. Convert to VHDL and also convert to Verilog. Now compare the code. Do I need to say more?

shared variables, guarded signals/blocks

I do not feel that anything can be eliminated

Not sure.

Ummm... guarded signals!

IN our VHDL design team, we don't use all the available constructs because it's not our choice or because they have not interest in modeling our systems. But I think it's the best way to keep a wide construct capabilities

No opinion.

21

Not much, if anything. Perhaps guarded signals (but not blocks). I don't feel that the complications are in the specific constructs, but in the language structure itself - look at section 10 of the LRM.

The ones I don't use...

Block statements Guarded signals Shared variables.

buffer ports linkage ports guarded things (signals/ports, blocks, GUARD signal) disconnection specifications replacement characters postponed processes groups next statement pulse rejection limits

complexity of expression language

with..select; disconnect; bus; block; extended identifiers; group; alternative delimiters;

there are probably some that I don't know about that could be eliminated but then if I knew about them, I might use them.

I don't see any need for the difference between ARCHITECTURE and ENTITY. At least I found no reason for re-using an entity by different architectures (in VHDL-AMS).

Some variations of "wait". "Guarded" may be eliminated when Wait is in one form.

Conditional continuous assignment could go since it can always be replaced by an equivalent process, but there are times when it's convenient.....

groups guarded signals shared variables

Guarded evaluation – Process sensitivity lists - User-defined enumerated types - Operator overloading

Guarded blocks, pulse rejection etc

I do not think VHDL has to be simplified

access types

The buffer ports

groups, linkage ports.

substitute special characters for keywords (like C)

None of the RTL Subset. Configuration Specifications If I want one specific binding, I use either default binding (RTL) or direct instantiations (Testbenches).

35. What VHDL constructs cause you the most annoyance (what are your "pet peeves")?

36. What improvements could be made to VHDL to improve modeling at the behavioral and lower levels, i.e., the synthesizable behavioral, RTL, gate level ?

37. What improvements could be made to VHDL to improve modeling at the higher levels, i.e., system-level, systems that include hardware, software, analog, MEMS, etc. ?

some kind of information hiding like in object oriented languages (not like in ADA, but more like in Object-C)

Not sure.

Relax the tide type checking, check the types during elaboration. Allow more flexible Interfaces for VHDL-AMS

The making of concepts used in high-level modelling, e.g. resources, primitives in the language. I do like the fact that VHDL+ has resources that have limited access and block when this limit is reached as this avoids the need to implement mechanisms like semaphores explicitly in the model.

Pass

A C++ API.

efficient dynamically memory allocation like the new-operator in C++ - garbage collection like JAVA - defining the range of monitored signal events – signal example : std_logic monitored 100 This means that always the last 100 events are monitored (time and value) – with more flexible attributes 'last_value(monitornumber) 'last_event(monitornumber) 'last_time(monitornumber) you can observe more signal informations inside the testbench - generic ports: equivalent to a signal declaration inside the architecture declaration but accessable outside ONLY for simulation purposes - allow unconstraint types inside a record - OO classes containing a private, public AND a generic section: class t_class generic: bit_width : natural := 8; public: type t_position is natural range 0 to 1024; type t_data is record dataword : std_logic_vector(bit_width-1 downto 0); dataposition: t_position; end record; function get_data(position) return std_logic; private: signal data : t_data; end class t_class; inside an architecture a signal can be defined: signal my_data : t_class(bit_widt => 10); signal one_bit : std_logic; then you have access to the members inside the public section of the class, e.g. get_data function one_bit <= my_data.get_data(4);

The same as above.

Automatic sensitivity list generation if absent

Some _simple_ analog extensions.

data entry tools (don't fix the language, if it ain't broke) VHDL is elegant, no one has any business writing it.

OOP (add inheritance to abstract data types)

systems that include hardware,sofeware

Analog, mixed, and MEMS should be covered by VHDL-AMS. However, HW/SW cosimulation should be investigated leading to new language capabilities and/or new packages.

In my opinion, a clean design of VHDL is necessary. Therefore backward compatibility should be preserved wherever possible, but readability and ease of use should be given higher precedence.

Support for software specification, simulation, and code generation (like SDL).

Object oriented features! Easier type conversions.

For testbench simulations, I think it will be well if VHDL language could call subroutines written with high level languages (C, C++, JAVA,PERL). This improvement needs that CAD tools are able to run co-simulations. In addition, for simulations, it would be well if internals signals could be visible at the top of a module (like in VERILOG language). It's much better for debug step

don't know

Real pointers. Taking addresses of signals and subprograms. A preprocessor. A foreign language interface. Massive improvements to TextIO. Simplification of binding and visibility rules. In fact, a magical transformation of VHDL into C with signals.

Better file I/O. Better text handling.

Cant really comment here.

abstract form of communication (message-passing on channels) dynamic process creation/termination

Polymorphism between entities. OOVHDL goes someway to this, but won't handle mapping between signal, terminal and quantity ports in VHDL-AMS. Sort out shared variables!!

Restrict synchronisation with clock to one construct only.

Classes and inheritance A PLI

An include statement like C's #include

Need to architect a bi-directional I/F with SLDL's constraints environment, to allow partitioning and budgeting of blocks based on budgeting of constraints. Need relaxation of the need for absolute-time clock definition for events. Need std I/F with C/C++ modeling (OMF sometimes not a good fit).

More construcys for manipulating data in files. Very helpful when running test benches.

Addition of proper textio functions. A function similar to printf in C would be great

Not really sure, but this area is very interesting to me at present.

see 36 - easier file I/O + interprocess communication - pointer types - easier type casting –

Hierarchical referencing is needed for testbenches. Typically I need to get at the value of a tristate enable for my design. I have no way to see an event on this object without creating a port (then I have two copies of my design, which is bad) or using a global signal (then I have to lie to my synthesis tool to keep it from seeing the global signal - besides this is ugly).

38. What capabilities could be added to VHDL to help with IP creation, protection, delivery, and reuse ?

common "encrypted" code which can be recompiled on different machines and is also understood by synthesis tools (if synthesis is required)

Not sure.

Define some standard scrambling technique, such, that only every language construct is represented on basic concepts (without structure or with some defined level of structure). (Similar to object code).

Don't use for this so no comment

pass

24

External model interfacing (another C++ API).

an IP interface for including binary coded IP cores combined with an coding key (this means that the simulator syntheziser only can use the IP core together with this key from the IP core developer) this key can contain any information needed to protect the IP core or to associate it to a specific user for a speific time (license...)

Good question for which I don't have an answer.

VHDL Byte code equivalent of Altera's Jam Byte Code => obfuscation.

Compiler directive for technology specific RPM's? This would be sweet for FPGA IP.

this is a vendor bugaboo

Standards for management of group development of IP. (i.e. Standards for revision control and distribution.)

Polymorphism applied to entity-architectures.

A way of including synthesis information (e.g. timiconstraints) if you want to deliver behavioural IP.

creation: can be done in VHDL - protection: should not be considered as an issue of VHDL - delivery: see protection - reuse: see creation

Support for object code delivery of IP models.

I couldn't say. I'm perfectly happy if someone hands me some IP that consists of some shrouded VHDL source files, and that strikes me as providing pretty reasonble protection for the IP vendor.

No opinion. To my mind, it's not important.

A standard "compiled" format that allows distribution of models without releasing source code. Some products have this in a proprietary way already.

I don't know, specifically. But I do know that I don't want to be forced to deliver source.

Computer readable comments, along the lines of the JAVA /** comment. Would help tool based use of IP.

Polymorphism between entities. OOVHDL goes someway to this, but won't handle mapping between signal, terminal and quantity ports in VHDL-AMS. Sort out shared variables!!

We do not wish to be tied to a particular editor. So we have not used any of the state machine/truth table editors. VHDL should be changed to better support entry of state machine descriptions. A table description would be useful.

Test collar encapsulation, e.g. TAP controller

Don't know

Use the SLDL coupling to create a "grey-box" model that defines the IP's function and constraints as viewed from the external ports (timing, power, area, whatever). For protection, perhaps leverage OMF further for seamless mix-match of (compiled) IP blocks for design verification. Make OMF model generation a common output feature for VHDL models, as in "Export to OMF for Delivery" or similar.

25

I don't see much of a need for these capabilities on top of the facilities VHDL already has.

More IP protection related. A few more reuse features also might help.

Addition of standard Synthesis controls specified in the VHDL source. These would specify timing and hierarchical control. if this was in in the standard then all Synthesis tools would support it

???

39. Are there other design challenges or issues that the survey failed to identify that you believe VHDL needs to address ?

not immediately

NO.

You should make a list which language constructs are necessary. And the participants have to flag them. Then you can find out which are used and which are not used.

No

Pass

The whole approach to signal resolution (and type conversion) should be reviewed in light of requirements for mixed-signal (multi-simulator) requirements, particularly analog/digital boundary handling, as the existing approach is semantically broken. However, as this is mostly a semantic rather than syntactic problem it should not be to hard to fix.

Perhaps

ANYTHING that would make the language less verbose would be welcome. Most of the designers I work with know 'C', I have never met anyone who knew 'ADA'. If Verilog had arrays and records VHDL would sit on the shelf like a dead raccoon.

VHDL is already quite complex, not realy easy to understand to newbies. If you add to many features, it may become unusable. Why not identify a "core" VHDL for pure synthesizable digital IC design and extensions for other usages (testbench, system-level specs, etc)?

no

no

no

I haven't thought of any in the last 10 minutes...

a standard syntax for compiler directives.

None I can think of

Perhaps better handling of timing and delay data (not SDF, but perhaps integrate DCL / OLA API support into VITAL and/or VHDL). Explicit handling of interconnect delays are particularly needed.

Not off hand.

I think that we should remember that there are also a lot of users out there doing small designs rather than the 1M gate ASICs', the language should be kept as simple as possible for them. The small FPGA market is very large and there are a lot of users out there who want basic tool sets and are not prepared to pay for tools that support many more features than they need such as VPLI. If the language does get extended we need to make sure that basic compliant toolsets can still be supplied

???

Sorry for the late reply. I would like to participate more. I have not read news groups for a long time. Please post this type of stuff to the VHDL reflectors. I am a member of SIWG. I am sure once I give this some more thought that I can add some more to my issues list. Is there a method for doing this? My email = Jim@SynthWorks.com The current language specification is very difficult to read and correctly understand. It would greatly help those of us who can read BNF if the BNF were exact rather than general. For example, the BNF should specify that functions may only have parameters of mode in. I realize that this is a huge effort; however, if the LRM is to be useful to anyone other than tool developers (read users - hardware designers, testbench developers), this is necessary.

Appendix 3

Survey methodology and survey form

VHDL-200x – VHDL for the New Millennium Survey Form

This is the first part of a two-part survey. This part of the survey is intended to gather general information from a large population of users to help compile a list of user needs that will be used to formulate requirements for the next update to the VHDL language (VHDL 200x).

The second part of the survey will be a more detailed interview concerning VHDL 200x requirements. Please indicate your willingness to participate in the second part of the survey by filling out the contact information requested in question 26.

Note that only a subset of those indicating willingness to participate in the second part of the survey will be contacted.

1. Which HDL do you use (check all that apply) ?

- ☐ VHDL
- ☐ Verilog
- ☐ SDL
- ☐ C
- ☐ C++
- ☐ Java
- ☐ VHDL-AMS
- ☐ VERILOG-AMS
- ☐ MAST or other Analog language
- ☐ Other (please specify) [   ]

2. VHDL is used in my organization for (check all that apply) ?

- ☐ Digital ASIC/IC
- ☐ FPGA design
- ☐ PCB
- ☐ Performance Modeling
- ☐ High-Level System Design
- ☐ RTL Design
- ☐ Gate Level Design
- ☐ Analog design
- ☐ Education
- ☐ IP core development
- ☐ System-on-a-chip design
- ☐ Other (please specify) [   ]

3. VHDL is <u>primarily</u> used in my organization for (select ONE ONLY) ?

- ○ Digital ASIC/IC design
- ○ FPGA design
- ○ PCB
- ○ Performance Modeling
- ○ High-Level System Design
- ○ RTL Design
- ○ Gate Level Design Analog design
- ○ Education
- ○ IP core development
- ○ System-on-a-chip design
- ○ Other (please specify) [_____]

4. I personally use VHDL for (check all that apply) ?

- ☐ Digital ASIC/IC design
- ☐ FPGA design
- ☐ PCB
- ☐ Performance Modeling
- ☐ High-Level System Design
- ☐ RTL Design
- ☐ Gate Level Design
- ☐ Analog design
- ☐ Education
- ☐ IP core development
- ☐ System-on-a-chip design
- ☐ Other (please specify) [_____]

5. My primary job responsibility is (check ONLY ONE) ?

- ○ Corporate management
- ○ Design engineering
- ○ Design engineering management
- ○ Engineering/technical support
- ○ Research and development
- ○ Technical marketing
- ○ EDA tool development
- ○ IP core development
- ○ Other (please specify) [_____]

6. Primary end application of my designs are (check all that apply) ?

- ☐ Avionics, Marine, Aerospace Electronics
- ☐ Communication systems
- ☐ Computers
- ☐ Computer peripherals
- ☐ Consumer electronics
- ☐ Automotive/Other Ground vehicles
- ☐ Industrial controls, Robotics
- ☐ Electronic Instrumentation/ATE/Design and Test
- ☐ Medical Electronics, appliances
- ☐ Educational institution
- ☐ Military
- ☐ Other (please specify) [_____]

7. Please estimate the relative percentage use of each capture method for your VHDL designs ?

- [0% ▼] Text Editor
- [0% ▼] Language-sensitive Editor
- [0% ▼] Schematic Editor
- [0% ▼] State machine/chart/table Editor
- [0% ▼] Truth Table Editor
- [0% ▼] Reused design models
- [0% ▼] Other (pleae specify) [_____]

8. If you are using VHDL synthesis, please estimate the percentage of your designs that are synthesized to each of the following?

- [0% ▼] VITAL gates
- [0% ▼] Other VHDL gate-level descriptions
- [0% ▼] Verilog gates
- [0% ▼] Other non-VHDL or Verilog gate level descriptions

How Important are the Following Capabilities to Your Future Success with VHDL

(scale – 1=not important -- 5=very important)

9. Hardware/Software co-design support ?

[1=not important ▼]

10.System-level constraint specification within VHDL (power, timing constraints, etc.) ?

| 1=not important ▼ |

11. Fault simulation with VHDL ?

| 1=not important ▼ |

12. System level modeling (e.g., performance and stochastic modeling) ?

| 1=not important ▼ |

13. High level synthesis (from behavioral models to RTL) ?

| 1=not important ▼ |

14. Support for new hardware design paradigms such as data flow and reconfigurable hardware ?

| 1=not important ▼ |

15. Accelerated simulation via cycle-based, synthesizable or other subset standardization ?

| 1=not important ▼ |

16. Accelerated simulation via more efficient interprocess communication mechanism (messages, shared variables, lightweight signals, etc.)?

| 1=not important ▼ |

17. Better encapsulation and abstraction capabilities ?

| 1=not important ▼ |

18. Better reuse capabilities ?

| 1=not important ▼ |

58

19. Better communications modeling capabilities ?

| 1=not important | ▼ |

20. Standard programming language interface by which externally generated (foreign, e.g. C, C++) models and functions can be included in the simulation of a design ?

| 1=not important | ▼ |

21. Support for externally generated models that conform to OMF and VSI emerging standards ?

| 1=not important | ▼ |

22. Support for IP creation, protection, delivery, and reuse ?

| 1=not important | ▼ |

23. A standard simulation control language ?

| 1=not important | ▼ |

24. A standard simulation waveform database format ?

| 1=not important | ▼ |

25. Ability to interface to high-level system description languages (SDL, SLDL) ?

| 1=not important | ▼ |

26. How important is it that the next update of VHDL be upward compatible (i.e., existing models work when executed by tools compliant to the new version) with previous versions of VHDL ?

| 1=not important | ▼ |

27. Are you aware of the following extensions to VHDL currently being pursued and do you feel they will be useful in addressing some of the capabilities described above ?

OOVHDL   O Yes   O No   O Not aware/unsure

SID (VHDL+)   O Yes   O No   O Not aware/unsure

28. Do you feel the extensions to VHDL currently being pursued will be useful to YOU in the future?

OOVHDL   O Useful   O Not useful   O Not aware/unsure

SID (VHDL+)   O Useful   O Not useful   O Not aware unsure

Essay Questions

29. Where in the design cycle (specification, high-level design, detailed design, implementation, testing, etc.) is the majority of time spent in your organization ?
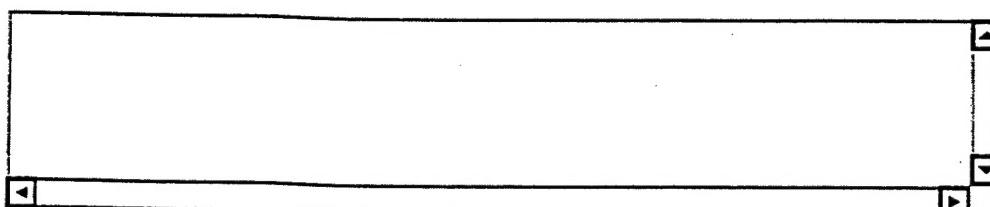
30. What is the current most important application area of VHDL ?
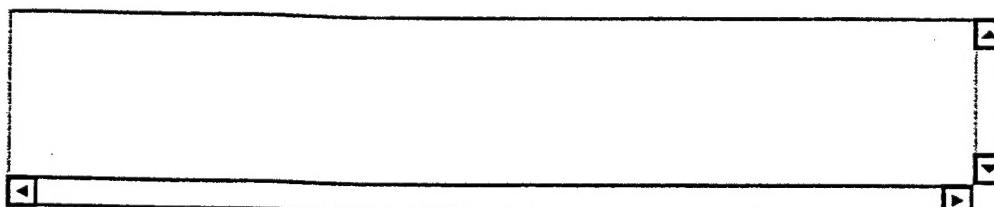
31. Are there any other areas where you use VHDL that were not covered by questions 2-4 ?
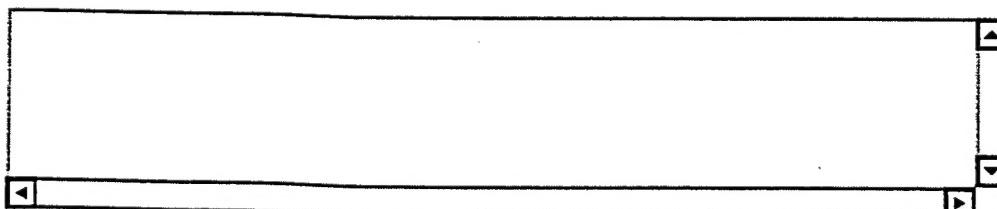
32. In designing the next update to VHDL, if a choice between language capability and readability/ease of use vs. backward compatibility has to be made for one or more constructs, would you prefer strict backward compatibility or less capability in the revised language?
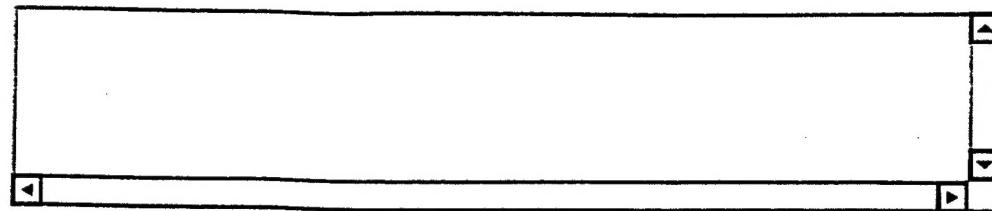
33. What design languages (including any from question 1 including VHDL and SLDL) do you plan to use in the next 2 years, for what purpose, and why the specific language?
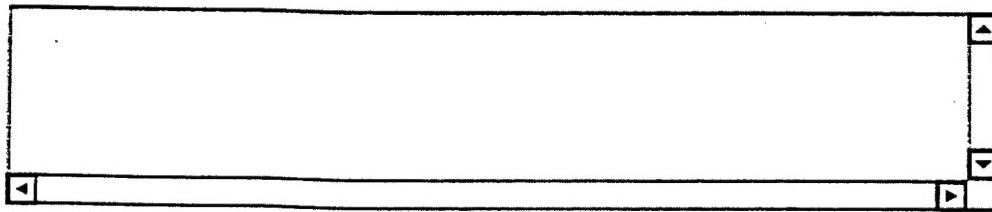
34. In the interest of simplifying VHDL what language constructs do you feel could be eliminated?
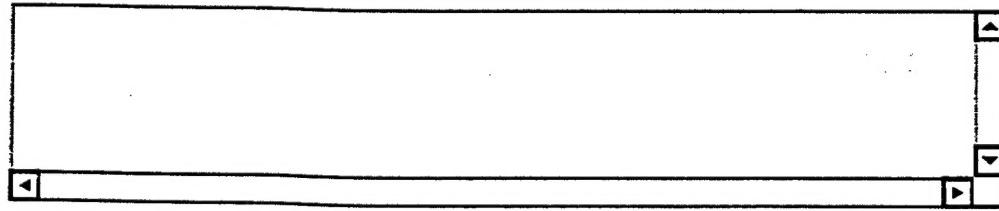
35. What VHDL constructs cause you the most annoyance (what are your "pet peeves")?
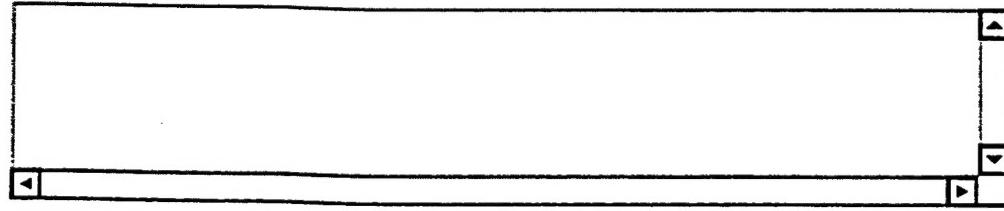
36. What improvements could be made to VHDL to improve modeling at the behavioral and lower levels, i.e., the synthesizable behavioral, RTL, gate level ?
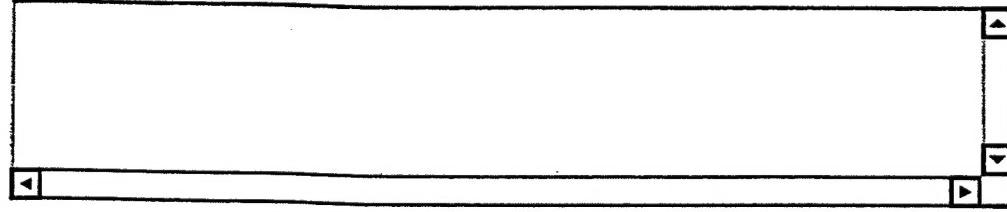
37. What improvements could be made to VHDL to improve modeling at the higher levels, i.e., system-level, systems that include hardware, software, analog, MEMS, etc. ?

38. What capabilities could be added to VHDL to help with IP creation, protection, delivery, and reuse ?

39. Are there other design challenges or issues that the survey failed to identify that you believe VHDL needs to address ?

40. (Optional) The survey's authors would like to be able to contact you to discuss your responses and your thoughts on VHDL 200x.

If you supply the information below, it will be used ONLY for that purpose, it will not be included in the survey data (with the exception of location data City and Country).

Name:

Company:

Email address:

Phone number:

City:

Country:

Submit This VHDL Survey Form   Reset